Algorithms and Static data structures

Serie of exercises 5

December 2024

Exercise 1: Circular Shift

Write an algorithm that performs a circular shift of the elements in an array to the right.
For example, given the array [1,2,3,4], after a circular shift, the array becomes [4,1,2,3].
Write an algorithm that rotates the elements of a one-dimensional array to the left by K positions.

Example:
Input Array: [1,2,3,4,5] K=2
Output: [3,4,5,1,2]

Exercise 2: Merge Two Sorted Arrays

Write an algorithm that merges two sorted one-dimensional arrays into a single sorted array.

Example:
Array 1: [1,3,5]
Array 2: [2,4,6]
Output: [1,2,3,4,5,6]

Exercise 3: Maximum Product Subarray

Write an algorithm to find the maximum product of a contiguous subarray within an array.

Example:
Input: [2,3,−2,4]
Output: 6 (The subarray is [2,3])

Exercise 4: Trap Rainwater

Given an array where each element represents the height of a bar, find the amount of water that can be trapped after rain.

Example:
Input: [0,1,0,2,1,0,1,3,2,1,2,1]
Output: 6

Exercise 5: Partitioning Around a Pivot

Write an algorithm to rearrange the elements of an array such that all elements less than a pivot appear before it, and all elements greater than the pivot appear after it. The pivot itself should be in its correct position.

Example:
Input: Array = [9,12,3,5,14,10,10] Pivot = 10
Output: [9,3,5,10,10,12,14]

Exercise 6: Subarray with Maximum Sum

Write an algorithm to find the contiguous subarray (containing at least one number) that has the maximum sum, and return its sum.
Example:
Input: [−2,1,−3,4,−1,2,1,−5,4]
Output: 6 (The subarray is [4,−1,2,1])

Exercise 7: Longest Increasing Subsequence

Write an algorithm that finds the length of the longest increasing subsequence in an array.
Example:
Input: [10,22,9,33,21,50,41,60,80]
Output: 6 (The subsequence is [10,22,33,50,60,80])

Exercise 8: Sorting by transposition

Two consecutive elements are compared and then swapped if the second element is smaller than the first. A backward check is then performed to verify if the order has been disrupted, in which case it is restored.

Exercise 9: Bubble sort

Implement the Bubble Sort algorithm to sort an array of integers in ascending order, compare adjacent elements, swap them if they are in the wrong order (i.e., if the left element is greater than the right element). Repeat the process until the array is sorted. Optimize the algorithm by stopping early if no swaps are made in a pass (indicating the array is already sorted).

Additionally, modify the algorithm to count the number of swaps performed during the sorting process.

Exercise 10: Counting sort with 3 arrays

It is done in 2 steps. In the first step, for each element, the number of elements smaller than it is counted and the result is stored in a counting table (in cells with the same index, of course) of the same size as the array to be sorted. In the second step, each element from the initial array is placed in its correct position (since we know the number of elements smaller than it) in the third table.

Exercise 11: Counting sort with 2 arrays

Counting sort with 2 arrays is performed in two steps: first, the counting array is constructed, where each element stores the number of elements smaller than it. In the second step, the counting array is traversed, and for each element, it is swapped with the element in its correct position in the array to be sorted, as the number of smaller elements is already known. It's important to also swap the elements in the counting array simultaneously with those in the array to be sorted. Special attention should be given to cases where there are duplicate elements in the array.

Exercise 12: Counting sort with one array

There is no counting array or result array. In other words, as soon as the number of elements smaller than an element is found, the element is placed in its correct position. However, care must also be taken with elements of the same value.

Exercise 13: Rotate a square matrix by 90°

Write an algorithm to rotate a square matrix by 90° clockwise.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \Longrightarrow \begin{bmatrix} 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \end{bmatrix}$$

Exercise 14: Matrix determinant (3x3)

Write a program to compute the determinant of a 3x3 matrix.

Example :

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 1 & 0 & 6 \end{bmatrix}$$

Determinant = 14

Exercise 15: Matrix chain multiplication

Write an algorithm to compute the optimal parenthesization of matrix chain multiplication.
Given a sequence of matrices, the program should output the minimum number of scalar multiplications needed to multiply all the matrices.
Input:
Matrices A (10x30), B (30x5), C (5x60), D (60x20)
Output:

Optimal multiplication order: ((AB)C)D
Minimum number of scalar multiplications = 4500

Exercise 16: Matrix inverse (3x3)

Write an algorithm to compute the inverse of a 3×3 matrix. If the matrix is not invertible (its determinant is zero), the algorithm should indicate this.

Input:

Matrix A = $\begin{bmatrix} 2 & 3 & 1 \\ 1 & 2 & 1 \\ 3 & 1 & 2 \end{bmatrix}$

Output:

Inverse of A = $\begin{bmatrix} 0.5 & -1 & 0.5 \\ 0.5 & 1 & -0.5 \\ -1.5 & 2 & -0.5 \end{bmatrix}$

Exercise 17: Pathfinding in a Matrix

Given a N×M matrix filled with integers, find the path from the top-left corner to the bottom-right corner with the minimum sum of values, moving only down or right at each step.

Example:

$$A = \begin{bmatrix} 1 & 3 & 1 \\ 1 & 5 & 1 \\ 4 & 2 & 1 \end{bmatrix}$$

Output: Minimum sum = 7.

Exercise 18: Removing Identical Rows Using a Pivot Matrix

Given two matrices A(m,n) and B(k,n) remove all rows from A that are identical to any row in B, but keep the rows whose **sum of elements** is strictly greater than a given threshold S.

$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}$

$B = \begin{bmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

$S = 20$

$A_{\text{filtered}} = \begin{bmatrix} 10 & 11 & 12 \end{bmatrix}$